

An abstract domain for separation logic formulae

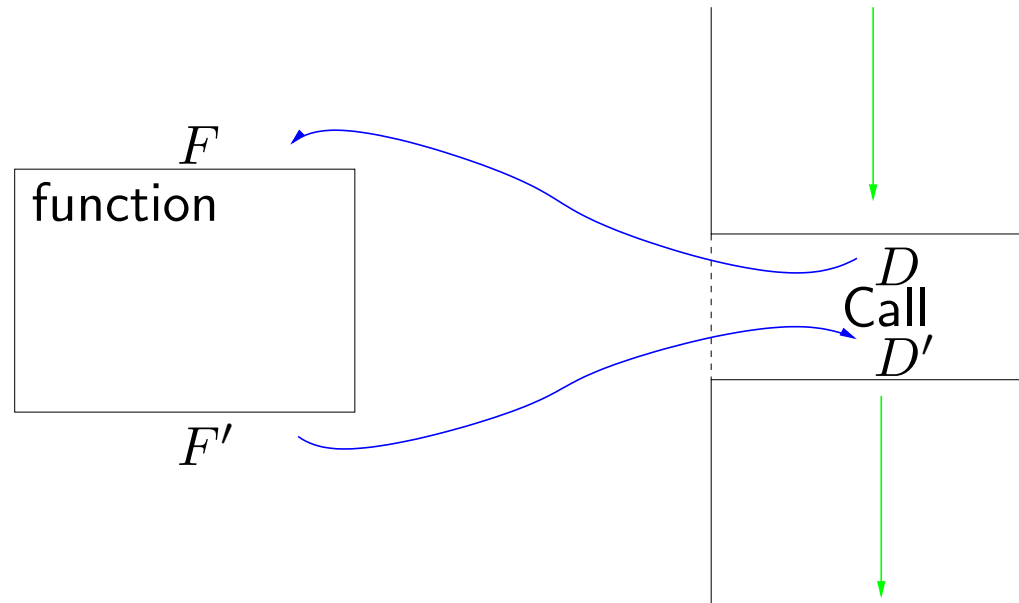
Élodie-Jane Sims

Elodie-Jane.Sims@polytechnique.fr

Situation

- ❖ Goal: **pointer analysis**: check dereferencing errors, aliases, ...
- ❖ $BI^{\mu\nu}$ a separation logic which permit easy descriptions of the memory, e.g.
 - x points to a list of [1;2;3]
 $\exists x_2, x_3. (x \hookrightarrow 1, x_2) * (x_2 \hookrightarrow 2, x_3) * (x_3 \hookrightarrow 3, \text{nil})$
 - x and y are aliased pointers
 $x = y \wedge \exists x_1, x_2. (x \hookrightarrow x_1, x_2)$
 - Partitionning: x and y belong to two disjoint parts of the heap which have no pointers from one to the other...

- We want to use this logic as an **interface** language for **modular analysis**



Analysis 1 $\rightarrow BI^{\mu\nu} \rightarrow$ Analysis 2

Program $\rightarrow BI^{\mu\nu} \rightarrow$ Analysis 3

We have build an intermediate domain such that:

- it is similar to the existing shape/alias analysis domains to allow translations from/to those domains
- it comes with a concrete semantics in term of sets of states which is the same domain as for the formulae's semantics
- we can translate the formulas into our domain
- it is a cartesian product of different subdomains so that we can cheaply tune the precision depending on the needs (for example, the domain is parametrised by a numerical domain which can be forgotten if we do not care about numericals)

The concrete domain: *State*

We have a set of variables Var .

$$\begin{array}{llll} Val & = & Int \cup Bool \cup Atoms \cup Loc & \text{Values} \\ S & = & Var \rightarrow Val & \text{Stacks} \\ H & = & Loc \rightarrow Val \times Val & \text{Heaps} \\ \textit{State} & = & S \times H & \end{array}$$

Rq: stacks can be partial functions

The logic: $BI^{\mu\nu}$

<i>Classical</i> <i>connectives</i>			
	$E = E'$		false
	$P \Rightarrow Q$		$\exists x.P$
<i>Spatial</i> <i>connectives</i>			
	emp		$E \mapsto E_1, E_2$ Points to
	$P * Q$		$P \multimap Q$ Spatial imp.
<i>Fixpoints</i> <i>connectives</i>			
	X_v		$P[E/x]$ Posponned substitution
	$\nu X_v.P$		$\mu X_v.P$ Least fixpoint

$Var_v = \{X_v, Y_v, \dots\}$ infinite set of variables of formulae

Semantic of $*$

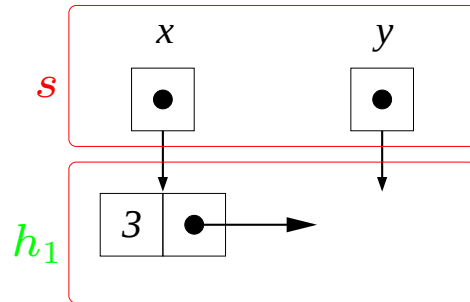
$$\llbracket P * Q \rrbracket_\rho = \left\{ s, h_0 \cdot h_1 \mid \begin{array}{l} \bullet \text{ } \textcolor{red}{dom(h_0) \cap dom(h_1) = \emptyset} \\ \bullet \text{ } s, h_0 \in \llbracket P \rrbracket_\rho \\ \bullet \text{ } s, h_1 \in \llbracket Q \rrbracket_\rho \end{array} \right\}$$

Examples of formulae

Ex. 1

$$s = [x \rightarrow l_1, y \rightarrow l_2]$$

$$h_1 = [l_1 \rightarrow \langle 3, l_2 \rangle]$$

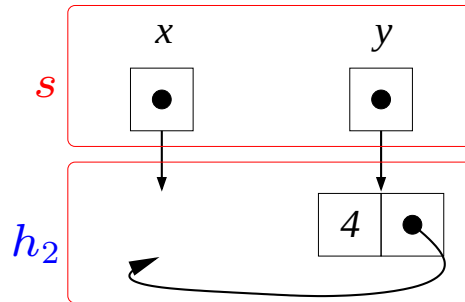


$$\models (x \mapsto 3, y)$$

Ex. 2

$$s = [x \rightarrow l_1, y \rightarrow l_2]$$

$$h_2 = [l_2 \rightarrow \langle 4, l_1 \rangle]$$

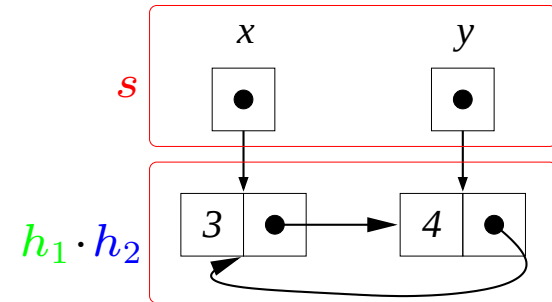


$$\models (y \mapsto 4, x)$$

Ex. 3

$$s = [x \rightarrow l_1, y \rightarrow l_2]$$

$$h_1 \cdot h_2 = \left[\begin{array}{l} l_1 \rightarrow \langle 3, l_2 \rangle, \\ l_2 \rightarrow \langle 4, l_1 \rangle \end{array} \right]$$



$$\models (x \mapsto 3, y) * (y \mapsto 4, x)$$

$$\not\models (x \mapsto 3, y) \wedge (y \mapsto 4, x)$$

Presentation of the choices for the domain by examples of translation of formulae

Ex1

Formulae	$x = \text{nil}$
Semantics	$\{s, h \mid s(x) = \text{nil}\}, \dots$
Translation	$\left(\boxed{x} \rightarrow \text{Nilt}, -, -, -, -, -, - \right)$

Formulae	$(x = \text{nil} \vee x = \text{true})$
Translation	$\left(\begin{array}{c} \boxed{x} \rightarrow \text{Nilt} \\ \boxed{x} \rightarrow \text{Truet} \end{array}, -, -, -, -, -, - \right)$

Ex2

Formula	$\neg(x = x)$
Semantic	$\{s, h \mid x \notin \text{dom}(s)\}$
Translation	$\left(\boxed{x} \rightarrow \text{Oodt}, -, -, -, -, -, - \right)$

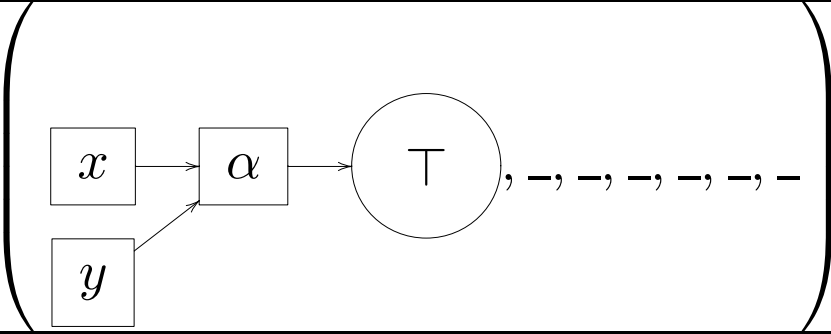
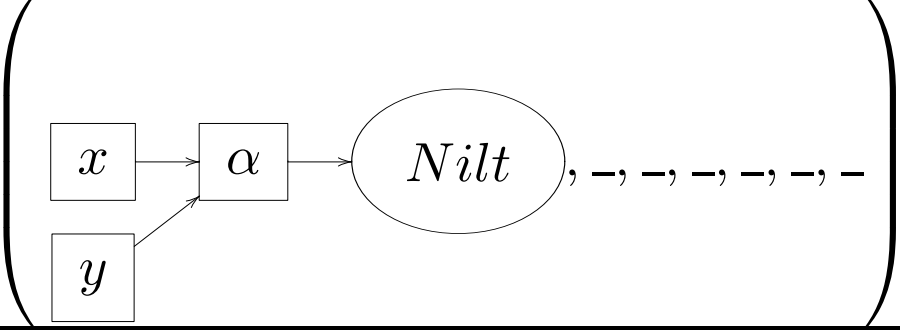
Do not confuse with:

Formula	false
Semantic	$\emptyset$
Translation (for ex.)	$\left(\boxed{x} \rightarrow \emptyset, -, -, -, -, -, - \right)$

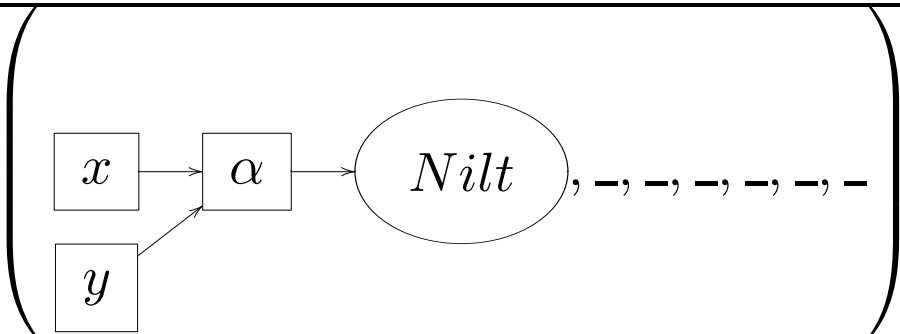
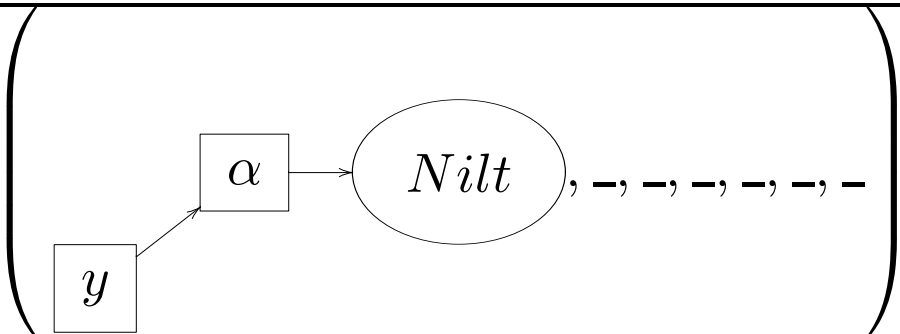
Ex3

Formula	$A \wedge B$
Constraints	cheap translation of $\wedge$
Translation	$T(A \wedge B) \triangleq T'(T'(\top, A), B)$

Ex4, Ex5

Formula	$x = y$
Constraints	refine the information for one variables while also refining the information of the second one in a cheap way
Adds	infinite set of auxiliary variables $TVar$ $VAR \triangleq Var \uplus TVar$
Translation	
Formula	$x = y \wedge x = \mathbf{nil}$
Translation	

Ex6

Formula	$x = y \wedge x = \text{nil}$
Translation	 The diagram shows a memory state enclosed in large parentheses. On the left, there are two boxes labeled x and y. Arrows from both x and y point to a box labeled α. An arrow from α points to an oval labeled Nil. To the right of the Nil oval, there is a sequence of six dashes: $, -, -, -, -, -, -$.
Formula	$(\exists x. x = y \wedge x = \text{nil}) \equiv (y = \text{nil})$
Translation	 The diagram shows a memory state enclosed in large parentheses. On the left, there is a box labeled y. An arrow from y points to a box labeled α. An arrow from α points to an oval labeled Nil. To the right of the Nil oval, there is a sequence of six dashes: $, -, -, -, -, -, -$.

Ex7

Formula	$(x < y + 3)$
Translation	$\left(\begin{array}{c} \boxed{x} \rightarrow \boxed{\alpha} \rightarrow \text{Numt} \\ \boxed{y} \rightarrow \boxed{\beta} \rightarrow \text{Numt} \end{array} , -, -, -, -, -, d \right)$ $d \in \mathcal{D} \text{ encodes that } \alpha < \beta + 3$

Ex8

Formula	"x is a location not allocated"
Semantic	$\{s, h \mid s(x) \in Loc \wedge s(x) \notin dom(h)\}$
Translation	$\left(\boxed{x} \rightarrow \text{Dangling_Loc}, -, -, -, -, -, - \right)$

Ex9

Formula	emp
Semantic	$\{s, h \mid \text{dom}(h) = \emptyset\}$
Adds	$\textcolor{blue}{HU} \triangleq \mathcal{P}(TVar)$ $\textcolor{blue}{HO} \triangleq \mathcal{P}(TVar) \uplus \text{full}$
Translation	$(-, -, \emptyset, -, -, -, -)$

Ex10

Formula	$(x \mapsto \text{true}, \text{nil})$
Semantic	$\{s, h \mid [s(x) \rightarrow \langle \text{True}, \text{nil} \rangle] = h\}$
Translation	

Formula	$(x \hookrightarrow \text{true}, \text{nil})$
Semantic	$\{s, h \mid [s(x) \rightarrow \langle \text{True}, \text{nil} \rangle] \subseteq h\}$
Translation	

Ex11

Variables represent at most one value. To allow approximation we introduce summary nodes which can represent several values.

Formula	$\text{approx. of } (x = \text{true} \wedge y = \text{false})$ $\text{by } \left(\begin{array}{c} x = \text{true} \\ \vee x = \text{false} \end{array} \right) \wedge \left(\begin{array}{c} y = \text{false} \\ \vee y = \text{true} \end{array} \right)$
Translation	

Ex12: finite acyclic list of *True* starting from *x*

Formula	$\mu X_v. \left(\begin{array}{l} (x = \text{nil}) \vee \exists x_2. \\ x \hookrightarrow (\text{true}, x_2) * X_v[x_2/x] \end{array} \right)$
Translation	

$\emptyset$ is the set of infinite summary nodes, for infinite list μ would be replaced by ν and $\emptyset$ by $\{\alpha\}$).

Ex13: increase precision of union

Formula	$\left(\begin{array}{l} x = \text{nil} \\ \wedge y = \text{true} \end{array} \right) \vee \left(\begin{array}{l} x = \text{true} \\ \wedge y = \text{nil} \end{array} \right)$
Translation	The diagram illustrates the translation of the formula into an abstract domain. It shows four nodes: α_1, α_2, α_3, and α_4. The variable x points to α_1 and α_2, while the variable y points to α_3 and α_4. The nodes α_1 and α_2 are connected to an abstract value $Nilt$, and α_3 and α_4 are connected to $Truet$. Dashed arrows labeled $\{\dagger_{eq}\}$ indicate relationships between α_1 and α_3, α_1 and α_2, α_2 and α_4, and α_3 and α_4.

Formal definition of the domain

$$\begin{aligned} VD1 &::= Numt \mid Truet \mid Falset \mid Oodt \mid Nilt \mid Dangling_Loc \mid TVar \\ VD &::= VD1 \mid Loc(\mathcal{P}(\{ *1, *2 \}) \times VD1 \times VD1) \\ PVD^+ &::= (\mathcal{P}(VD) \uplus \otimes, \sqcup, \sqcap) \end{aligned}$$

$$AD ::= VAR \xrightarrow{total} PVD^+$$

$$CL_{eq} ::= \mathcal{P}(\{ \dagger_{eq}, \dagger_{eq}, =_{eq}, \subset_{eq}, \supset_{eq}, \#_{eq}, \bigcirc_{eq} \})$$

$$TB ::= (TVar \times TVar) \xrightarrow{total} CL_{eq}$$

$$\begin{aligned} AR &::= AD \times \mathcal{P}(TVar) \times (\mathcal{P}(TVar) \uplus \mathbf{full}) \times \mathcal{P}(TVar) \times \mathcal{P}(TVar) \times TB \\ &\quad \times (\mathcal{D}, \llbracket \cdot \rrbracket^{\mathcal{D}} : \mathcal{D} \rightarrow (TVar \xrightarrow{total} \mathcal{P}(\mathbb{Z}))) \end{aligned}$$

Semantic

$$\begin{array}{ll} Val & \triangleq \mathbb{Z} \uplus Bool \uplus \mathbf{nil} \uplus Loc & Val' & \triangleq Val \cup \{\mathbf{ood}\} \\ S & \triangleq Var \multimap Val & S' & \triangleq Var \xrightarrow{total} Val' \\ H & \triangleq Loc \multimap (Val \times Val) & F & \triangleq TVar \xrightarrow{total} \mathcal{P}(Val') \\ & & R & \triangleq Loc \multimap \mathcal{P}(Loc) \\ State & \triangleq S \times H & MFR & \triangleq \mathcal{P}(S' \times H \times F \times R) \end{array}$$

Why the F ?

We want $\left[\begin{array}{c} x \rightarrow \alpha \rightarrow Numt \\ y \rightarrow \alpha \end{array} \right]^4 = \left[x \rightarrow \alpha \right]^4 \cap \left[\begin{array}{c} \alpha \rightarrow Numt \\ y \end{array} \right]^4 .$

$$\begin{aligned} \llbracket \cdot \rrbracket &\in \mathbf{AR} \rightarrow \mathcal{P}(\mathbf{State}) \\ \llbracket ar \rrbracket &\triangleq \{\bar{s}, h \mid s, h, f, r \in \llbracket ar \rrbracket'\} \end{aligned}$$

$$\begin{aligned} \llbracket \cdot \rrbracket' &\in \mathbf{AR} \rightarrow \mathbf{MFR} \\ \llbracket (ad, hu, ho, sn, sn^\infty, t, d) \rrbracket' &\triangleq \llbracket ad \rrbracket^4 \cap \llbracket hu \rrbracket^1 \cap \llbracket ho \rrbracket^{1'} \cap \llbracket sn \rrbracket^2 \cap \llbracket sn^\infty \rrbracket^{2'} \\ &\quad \cap \llbracket t \rrbracket^3 \cap \llbracket d \rrbracket^7 \cap \mathit{sem*} \end{aligned}$$

$$\begin{aligned} \llbracket \cdot \rrbracket^4 &\in \mathbf{AD} \rightarrow \mathbf{MFR} \\ \llbracket ad \rrbracket^4 &\triangleq \bigcap_{v \in \mathbf{VAR}} \llbracket v, ad(v) \rrbracket^5 \end{aligned}$$

Operations

- union
- extension (replace $[v \rightarrow S]$ by $[v \rightarrow \{v'\} | v' \rightarrow S]$ with a fresh v')
used to tune the precision of the union
- merging (replace $[v_1 \rightarrow S_1 \mid v_2 \rightarrow S_2]$ by $[v_2 \rightarrow (S_1 \cup S_2)]$)
used with the widening
- translations from formulae to the domain

Comparisons

- the $\bullet \begin{array}{c} 1 \\ \sim \\ 2 \end{array}$ represent locations in the usual shape graphs
- summary nodes as for other shape graphs, seems to give more possibilities than predicate abstraction (with each time a specific predicate for list, etc...) but the technics of predicate and their algorithm/heuristics (like folding/unfolding) could probably also be use on our graphs
- a lonely outgoing edge can be seen as a “must” arrow (or valued 1), several outgoing edges from a variable can be seen as a “may” arrow (or valued 1/2, but it is a bit more precise because we know that one of them should exist), and an edge to $\emptyset$ can be seen as a “must not” arrow (or valued 0)

- we deal with numerical (for what we know, only *Magill & al.* also do)
- we have a formal semantic of our domain, the semantics of auxiliary variables are formally defined and formally used in the proofs
- we don't have to check for equalities of variables
- the domain is a cartesian product, we can add or remove some parts depending on the precision we want

About translating *

We change a bit our domain so that :

$$\begin{aligned} VD1 &::= Numt \mid Truet \mid Falset \mid Oodt \mid Nilt \mid TVar \\ VD &::= VD1 \mid \textcolor{red}{Loct} \mid \textcolor{red}{Dangling_Loc} \mid Loc(\mathcal{P}(\{ *1, *2 \}) \times VD1 \times VD1) \end{aligned}$$

Restrictions: We present a translation $*$ with a function $*$ on elements of our domain with disjoint sets of used auxiliary variables.

The function sometimes return Ω as the error value.

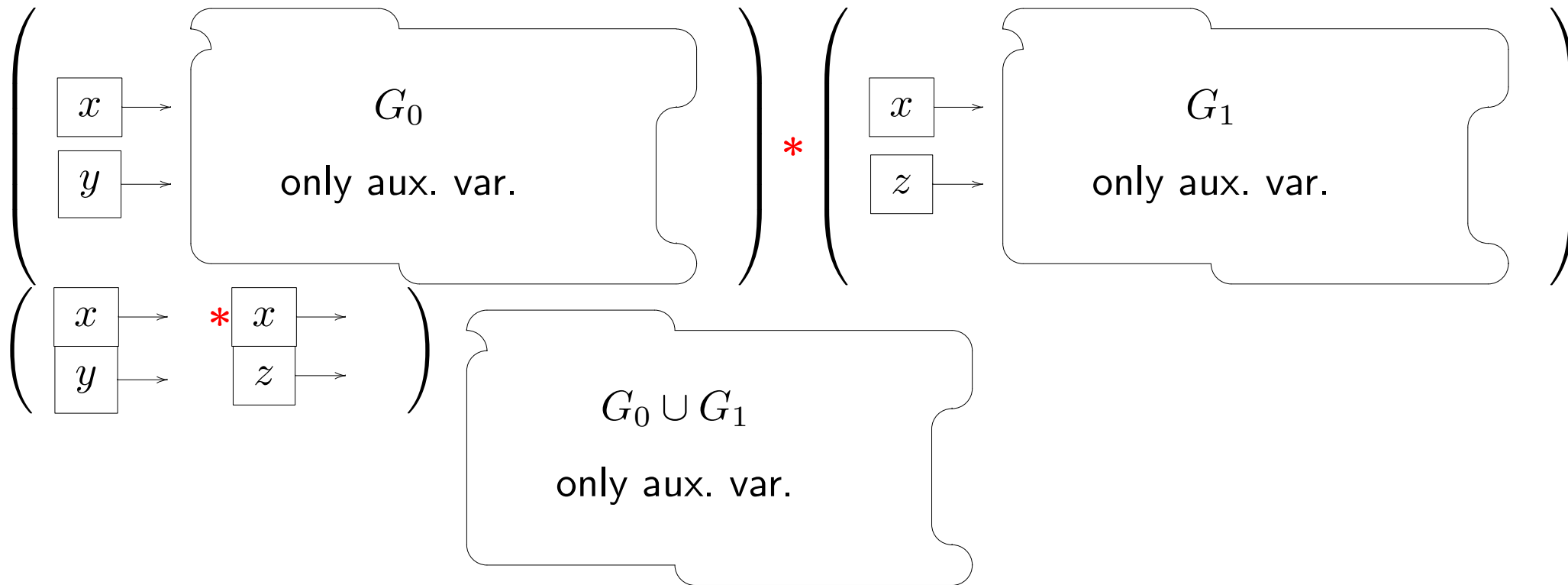
This is not in the article, here, we present only the key ideas of the algorithm.

* for simple types (no variables)

When $A, B \in \{Nilt, Truet, Falset, Oodt, Numt\}$, $A \neq B$

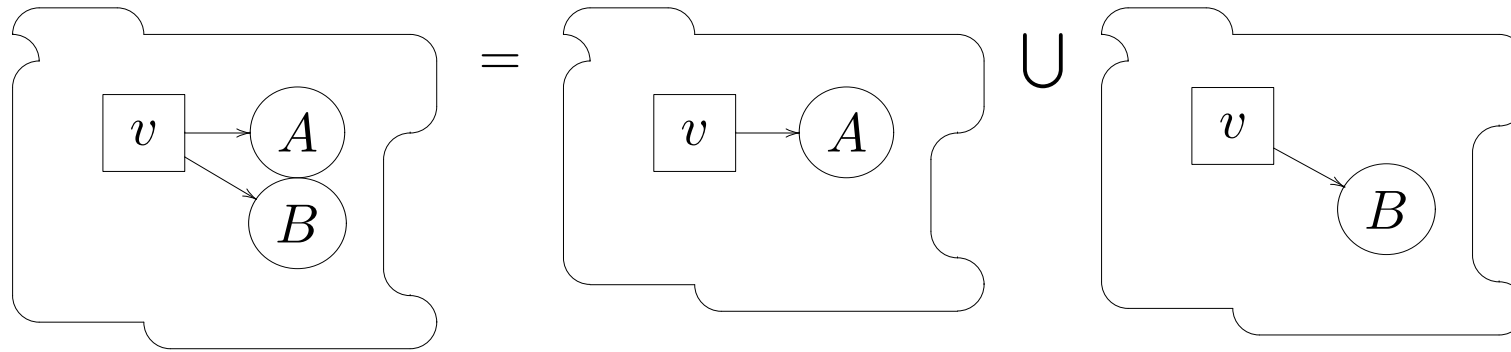
*	$\{A\}$	$\{B\}$	$\{Dgt\}$	$\{Loct\}$	$\{Loc(\dots)\}$	$\top$
$\{A\}$	$\{A\}$	Ω	Ω	Ω	Ω	$\{A\}$
$\{Dgt\}$			$\{Dgt\}$	$\{Loct\}$	$\{Loc(\dots)\}$	$\{Dgt, Loct\}$
$\{Loct\}$				Ω	Ω	$\{Loct\}$
$\{Loc(\dots)\}$					Ω	$\{Loc(\dots)\}$
$\top$						$\top$

The initial graph of auxiliary variables



Since $\text{dom}(G_0) \cap \text{dom}(G_1) = \emptyset$.

From graph of sets to sets of graph



For v variables (auxiliary or not) not summary nodes.

“Jumping” variables (simplified)

➤ α not summary node

$$\begin{aligned}
 & \left(\boxed{x} \rightarrow \boxed{\alpha} \rightarrow \bigcirc A \right) * \left(\boxed{x} \rightarrow \bigcirc B \right) \\
 = & \boxed{x} \rightarrow \boxed{\alpha} \rightarrow \left(\boxed{\alpha} \rightarrow \bigcirc A * \boxed{x} \rightarrow \bigcirc B \right) \\
 = & \boxed{x} \rightarrow \boxed{\alpha} \rightarrow \bigcirc A * B
 \end{aligned}$$

Because $\boxed{x} \rightarrow \boxed{\alpha}$ means $s(x) = f(\alpha)$ so the $*$ -information about x can be treated for α instead

➤ α cycles

$$\begin{aligned}
 & \left(\boxed{x} \rightarrow \boxed{\alpha_0} \rightarrow \dots \rightarrow \boxed{\alpha_n} \right) * \left(\boxed{x} \rightarrow \bigcirc B \right) \\
 = & \left(\boxed{x} \rightarrow \boxed{\alpha_0} \rightarrow \dots \rightarrow \boxed{\alpha_n} \rightarrow \bigcirc \top \right) * \left(\boxed{x} \rightarrow \bigcirc B \right) \\
 = & \left(\boxed{x} \rightarrow \boxed{\alpha_0} \rightarrow \dots \rightarrow \boxed{\alpha_n} \rightarrow \bigcirc \top * B \right)
 \end{aligned}$$

$s(x) = f(\alpha_0) = \dots = f(\alpha_n) = f(\alpha_0)$ is $s(x) = f(\alpha_0) = \dots = f(\alpha_n)$

► α summary node

$$\left(\begin{array}{c} x \rightarrow \alpha \rightarrow A \\ \searrow \\ B \end{array} \right) \neq \left(\begin{array}{c} x \rightarrow \alpha \rightarrow A \end{array} \right) \cup \left(\begin{array}{c} x \rightarrow \alpha \rightarrow B \end{array} \right)$$

but

$$\left(\begin{array}{c} x \rightarrow \alpha \rightarrow A \\ \searrow \\ B \end{array} \right) \subseteq \left(\begin{array}{c} \alpha \rightarrow A \\ \searrow \\ B \end{array} \right) \cup \left(\begin{array}{c} \alpha \rightarrow A \\ \searrow \\ B \end{array} \right)$$

Because $s(x) \in f(\alpha) \subseteq (A \cup B) \Rightarrow (s(x) \in A \wedge f(\alpha) \subseteq (A \cup B)) \vee (s(x) \in B \wedge f(\alpha) \subseteq (A \cup B))$

For example, we get

$$\begin{aligned}
& \left(\begin{array}{c} x \rightarrow \boxed{\alpha} \xrightarrow[*2]{\sim} \bullet \xrightarrow{1} \text{Truet} * x \rightarrow \text{Dgt} \\ \quad \searrow \text{Nilt} \end{array} \right) \\
= & \left\{ \begin{array}{c} \begin{array}{c} x \rightarrow \text{Nilt} \\ \boxed{\alpha} \xrightarrow[*2]{\sim} \bullet \xrightarrow{1} \text{Truet} \\ \quad \searrow \text{Nilt} \end{array} * x \rightarrow \text{Dgt} \end{array} \right\} \\
\cup & \left\{ \begin{array}{c} x \xrightarrow[*2]{\sim} \bullet \xrightarrow{1} \text{Truet} * x \rightarrow \text{Dgt} \\ \boxed{\alpha} \xrightarrow[*2]{\sim} \bullet \xrightarrow{1} \text{Truet} \\ \quad \searrow \text{Nilt} \end{array} \right\} \\
= & \left\{ \begin{array}{c} x \xrightarrow[*2]{\sim} \bullet \xrightarrow{1} \text{Truet} \\ \boxed{\alpha} \xrightarrow[*2]{\sim} \bullet \xrightarrow{1} \text{Truet} \\ \quad \searrow \text{Nilt} \end{array} \right\}
\end{aligned}$$

Ende